

# 高次元リーマン $\theta$ 関数計算におけるブロック対角構造の活用： $S(k,k)$ 因子化基盤の構築と計算工学的実効性の検証

Masaru Moriyama 独立研究家 (Independent Researcher)

2026 年 5 月 技術ノート／計算工学報告

## 概要

リーマン  $\theta$  関数の数値計算は「次元の呪い」を受ける：計算量は  $O((2N+1)^g)$  で次元  $g$  とともに指数的に増大し、一般の周期行列に対する **exact** 計算は通常  $g \leq 10$  程度に制限される。

本研究の核心は次の観察にある：本手法の基盤となる因子化恒等式は既知の数学的事実であり、その証明は定義から直接従う。にもかかわらず、この恒等式を体系的に実装してベンチマーク基準として整備した事例は著者が確認した範囲では存在しない。この空白を埋めることが本研究の目的である。この主張に反する事例をご存知の読者からの情報を歓迎する。近似手法が複雑な誤差解析によって精度を保証しなければならないのに対し、本手法は  $S(k,k)$  上での正確性を代数的恒等式から理論上は丸め誤差以外の誤差源が存在しない。この性質が本手法をベンチマーク基準として信頼できるものにする理由であり、体系的な実装が見送られてきた背景の詳細は 2.1 節で整理する。

さらに本研究では、計算ライブラリ FLINT (Elkies-Kieffer 2025) が、周期行列  $\Omega$  が厳密に **block-diagonal** ( $S(k,k)$ ) である場合に因子化恒等式を活用した再帰的次元削減を実装していないことを確認し、実測比較を行った (3.5 節)。(注：symplectic 行列の **block-diagonal** 分解機能は一部存在するが、本手法のような専用最適化は含まれていない。)

因子化恒等式自体は既知の事実であり証明は定義から直接従う。しかし次元パディング・最適  $k$  選択・ $S(k,k)$  近傍での誤差の定量化・打ち消し効果の理論化という一連の問題は非自明であり、本研究はその出発点を提供する。

本手法は「任意の周期行列」に対する汎用解法を目指すものではない。むしろ、物理学や幾何学の特定の文脈で自然に現れる  $S(k,k)$  構造 (**block-diagonal**) に特化することで、超高次元領域における **exact** な基準値を提供するものである。これは定理の証明ではなく、既知の代数的事実を極限 ( $g=20,000$  等) まで活用した『計算インフラ』の報告である。ただし本手法は、特定の構造クラス  $S(k,k)$  に特化した

「インディーカー的」最適化である—特定のコースで圧倒的な性能を発揮するが、任意のコースを走ることは目指していない。一方、FLINT は任意の周期行列に対する「F1 マシン」的汎用解法を目指している。両者は競合関係ではなく、むしろ相補的である。本手法は  $S(k,k)$  構造クラス限定で性能を発揮する一方、FLINT は優秀な汎用解法であり、任意の周期行列に対する厳密性・証明可能性を重視する。この対比は、構造特化アルゴリズムと汎用アルゴリズムの設計思想が生むトレードオフを示す好例である。

## 1. 背景と動機

リーマン  $\theta$  関数は  $\theta(z|\Omega) = \sum_{n \in \mathbb{Z}^g} \exp(\pi i n^T \Omega n + 2\pi i n^T z)$  と定義される ( $\Omega \in H_g$  は正定値虚部を持つ周期行列)。可積分系・代数幾何・数論・非線形信号処理など広範な数学・物理に現れる。

計算上のボトルネックは naive な格子和の指数的増大： $b=2N_{\text{cut}}+1$  として項数  $O(b^g)$ 。  $N_{\text{cut}}=1, g=22$  では約  $3.1 \times 10^{10}$  項となり、シングルスレッド実装 (AMD Ryzen 5 5500GT、Julia 1.12.6) で約 2.8 時間を要する。

実務的空白：

高次元  $\theta$  関数の近似手法 (Chimmalgi-Wahls 2023 等) は  $g=10$  以上で独立した exact 基準値による検証ができない。

特定の物理的・幾何学的文脈で  $g=17$  等の高次元が自然に現れるが既存手法では計算困難。

最先端ライブラリ FLINT に一般的な symplectic reduction での block-diagonal 検出機能は存在する (3.5 節で詳述)。

## 2. 既存研究における実務的空白

手法	実用上の最大 $g$	精度	exact ベンチマーク	精度保証の根拠	block-diagonal 最適化
Deconinck et al. (2004)	$g \leq 10$ 程度	任意精度	自身が基準	打ち切り誤差解析	なし
Chimmalgi-Wahls (2023)	$g \leq 60$ (近似、HPC 環境)	相対誤差 $\approx 0.01$	$g=10$ 以上なし	テンソル近似理論	言及のみ
FLINT/Elkies-	任意次元	任意精度	$g$ 方向は	区間演算	$S(k,k)$ 専用因子

Kieffer (2025)	(精度方向)		指数的		化・再帰的未実装
本手法 ( $S(k,k)$ )	任意次元 (構造クラス内)	機械精度	浮動小数点演算の範囲での <b>exact</b> 提供可	代数的恒等式 (定義から直接)	中核的設計

「精度の根拠」列に注目されたい。近似手法はいずれも非自明な誤差解析を要するが、本手法の精度は定義から直接従う代数的恒等式に基づく。独立した誤差解析を必要としない点がベンチマーク基準としての信頼性を高める。

なお、FLINT には現時点で  $S(k,k)$  構造に特化した再帰的因子化は実装されていない。これは任意の周期行列を広く扱うライブラリ設計における自然な選択だと考えている。本研究は、限定された構造に徹底的に最適化することで得られる性能を、別軸のアプローチとして提示するものである。

## 2.1 なぜこの空白が生まれたか：構造的要因

**block-diagonal** 構造を利用した **exact** 計算は数学的には定義から直接従うにもかかわらず、体系的な実装やベンチマーク基盤としての整備はこれまで行われてこなかった。この「理論的には単純だが、実装・検証の体系化には手間がかかるため、空白が生じていた」状況は、以下の四つの要因が重なっていた。

### (1) 研究文化の非対称性：一般解志向による特殊構造への注力の分散

主要研究 (Deconinck 2004・CW2023・FLINT 2025) はいずれも「任意の  $\Omega$  を扱う」ことを前提とし、特殊構造を前提とした **high-speed** 化は研究対象になりにくかった。 $S(k,k)$  のような特殊構造に基づく **exact** 計算は研究文化の外側に置かれたままになっていた。

### (2) 実務的需要の見落とし：特殊構造は「現場」で自然に現れる

可積分系・Hitchin 系・代数幾何の具体的応用では、 $g=17$  (GL(4)-Hitchin 系) のように **block-diagonal** 構造が自然に現れる場合がある。しかし近似手法 (CW2023 等) も  $g=10$  以上で独立した **exact** 基準値を持たず、「必要とされているが誰も提供していない」状態が続いていた。

### (3) 証明の単純さゆえの未実装：価値判断のギャップ

因子化恒等式の証明が定義への代入から直接従うため、数学者には数学的関心の対象が理論的性質に偏る傾向がある。しかし計算工学の観点では、独立した誤差解析を不要とする精度保証こそが **exact** ベンチマークとしての最大の強みである。「単純だから誰でもできる」と「誰も実装していない」の間に評価軸の相違を生んだ。

#### (4) 一般解向けライブラリ的设计思想との不整合

FLINT を含む既存ライブラリは「任意  $\Omega$  を扱う」設計思想を持つため、**block-diagonal** 構造の検出・因子化を前提にしていない。構造を利用すれば指数的高速化が可能であるにもかかわらず、構造依存の最適化が設計範囲に含まれていないというミスマッチが生じていた。

以上の四つの複合的要因が重なってこの空白は生まれた。本研究はこの要因によって生じた空白を埋め、**S(k,k)** 構造に対する体系的 **exact** ベンチマーク基盤を提供する。

#### (5) 類似事例：量子情報理論における同構造の空白（坂下 2013）

坂下（2013）[Sak2013] は密度行列のテンソル積  $A^{\otimes n}$  のブロック対角化という既知の代数的事実を量子 i.i.d. 状態の数値計算に系統的に応用し、論文自身が「実現可能・有効とも考えられていなかった」と明記している。本稿との共通点・相違点を以下に整理する。

項目	坂下（2013）	本手法
代数的事実	既約分解によるブロック対角化	<b>block-diagonal</b> 周期行列の因子化恒等式
削減の方向	テンソル次数 $n$ 方向	次元 $g$ 方向
削減の質	指数 $\rightarrow$ 多項式 $O(n^3)$ （質的变化）	$O(b^g) \rightarrow O(b^{(g/2)})$ （底の変化、指数のまま）
必要精度	多倍長演算必要（固有値分解の不安定性）	倍精度で十分（代数的恒等式により保証）
主目的	既存理論の数値検証	実装すること自体が空白を埋める
共通点	(a) 既知の代数的事実の数値実装が未整備 (b) 代数的構造で計算量削減 (c) 精度保証が恒等式から直接得られる (d) 文化的空白が存在した	

この事例は「定義から直接従う代数的事実の数値実装が空白になる」という現象が分野横断的な普遍的パターンであることを傍証する。両手法の組み合わせ可能性は確認していないが、今後の課題として記録する。

## 3. 手法

### 3.1 因子化恒等式：自明性が保証する精度

#### 命題 1（因子化恒等式）

$g_1, g_2 \geq 1$  とし、 $\Omega_1 \in H_{\{g_1\}}$ 、 $\Omega_2 \in H_{\{g_2\}}$  とする。 $\Omega = \text{diag}(\Omega_1, \Omega_2) \in H_{\{g_1+g_2\}}$ 、 $z = (z_1, z_2)^T \in C^{\{g_1+g_2\}}$  とする。このとき、

$$\theta(z | \Omega) = \theta(z_1 | \Omega_1) \cdot \theta(z_2 | \Omega_2)$$

が成立する。

証明：  $\theta$  関数の定義式に代入する。 $\Omega = \text{diag}(\Omega_1, \Omega_2)$  より指数部が  $n_1^T \Omega_1 n_1 + n_2^T \Omega_2 n_2$  と直交分解し、格子和が独立に因子化することから直接従う。  $\square$

(この等式は近似ではなく代数的に厳密である。定義から直接導かれるため、精度保証に独立した誤差解析は不要。)

本命題は既知の事実であるが、次元パディング戦略の正当性はこの命題から直接従う(3.2 節)。計算量は  $O(b^g)$  から  $O(b^{(g/2)})$  へと削減。 $N_{\text{cut}}=1, g=20$  では理論比  $3^{10}=59,049$ 、実測値  $58,525\times$ 。

### 3.2 次元パディング：FFT ゼロパディングの類比

本手法の工学的貢献は、 $g$  を  $S(k,k)$  構造が利用可能なサイズへ意図的に拡張する「次元パディング」の明示的提唱である。FFT 計算におけるゼロパディング ( $N$  を FFT フレンドリーなサイズへ拡張) と同じ論理構造を持つ。次元パディングは中間計算手段であり、命題 1 の因子化恒等式により最終的に得られる値は元の  $\theta(z|\Omega)$  に代数的に等しい。 $A_{\{g+\text{pad}\}}$  上での計算は元の  $\theta(z|\Omega)$  の値を代数的に正確に再現する。

	FFT (ゼロパディング)	本手法 (theta 次元パディング)
数学的事実	DFT の線形性 (定義から直接)	theta 関数の因子化 (命題 1、定義から直接)
精度保証の性質	独立した誤差解析不要	独立した誤差解析不要
工学的プロトコル (または手法)	$N$ を FFT フレンドリーなサイズへ拡張	$g$ を $S(k,k)$ フレンドリーなサイズへ拡張
明示的提唱の先例	Cooley-Tukey (1965)	本手法 (筆者が調査した範囲では先行例未確認)

DFT の線形性は既知であったが、Cooley-Tukey (1965) の工学的体系化は landmark contribution となった。数学的事実の証明が単純であることは計算工学的価値を損なわない。むしろ独立した誤差解析を必要としない精度保証は実装の信頼性を高める(2.1 節(3)参照)。

### 3.3 暗号理論との対比：代数的構造の逆用

本手法が対象とする  $S(k,k)$  構造 (block-diagonal 周期行列) は、暗号理論においては「分解可能な Jacobian」として知られ、離散対数問題の安全性を損なう分解可能性’と見なされる。具体的には、アーベル多様体の代数的分解に伴い、高次元の計算問題が低次元の部分問題へと可約 (Reducible) になる性質である。本稿の工学的貢献は、この既知の分解可能性’を、計算困難な高次元 theta 関数計算を効率化するための「計算量的な優位性を持つ部分空間 (Computational Locus)」として捉え直し、計算資源の最適化へと逆用した点にある。暗号では ‘分解可能性’ が安全性を損

なうため、 $S(k,k)$  構造を積極的に利用することはない。応用先は可積分系・Hitchin 系に限定される。

### 3.4 再帰的対数分解 (RLD)

$S(2,2)$  分解を再帰的に適用することで  $\log_2(g)$  段階の分解が可能。高次元オーバーフロー対策として対数空間演算 ( $\log(\theta_1 \times \theta_2) = \log(\theta_1) + \log(\theta_2)$ ) を採用。次元パディング・並列化・再開機能を実装した。これは計算工学的な実装面での工夫であり、数学的正当性は命題 1 から理論上は丸め誤差以外の誤差源が存在しない。

### 3.5 FLINT との比較

FLINT は 厳密な block-diagonal 周期行列  $\Omega$  に対する因子化恒等式  $\theta(z|\Omega) = \theta(z_1|\Omega_1) \cdot \theta(z_2|\Omega_2)$  を活用した再帰的対数分解および次元パディングを実装していない (2.1 節(4)参照)。symplectic 行列に対する block-diagonal 分解 (`sp2gz_is_block_diag`, `sp2gz_decompose`) および一部の dimension-lowering (`acb_theta_ql_lower_dim`) は実装されているが、本手法のような高次元 ( $g \gg 10$ ) での exact 機械精度対応の専用最適化は含まれていない。ソースコード確認 (FLINT `acb_theta` モジュール) および実測によりこの設計思想の違いが確認された。

本稿では FLINT と本手法の実測比較を行い、設計思想の違いを記録する。  
なお、実測比較 (2 環境: Intel N97 16GB・AMD Ryzen 7 5700X 64GB、Windows、Python-FLINT 0.8.0) の結果を以下に示す。両環境で定性的に同一の挙動が確認された。

注: FLINT (Elkies-Kieffer 2025, arXiv:2505.22382) では `sp2gz_decompose` 等による symplectic 分解は実装されているが、著者が確認した範囲では、 $S(k,k)$  構造に特化した再帰的因子化は実装されていないように見える。

g	Naive (s)	RLD (s)	FLINT Ryzen7 (s)	FLINT N97 (s)	Naive-RLD	RLD-FLINT
2	0.000	0.0001	0.038	—	0.00e+00	3.65e-11
4	0.001	0.0002	0.036	—	1.08e-19	2.83e-11
6	0.006	0.0002	34.47	42.8	1.11e-15	2.38e-11
8	0.053	0.0003	実行中(>7h)	29,205	4.88e-15	—
10	—	0.0007	— (nan)	nan	—	—
12	—	0.0009	— (OOM)	OOM	—	—

これは設計思想の差を反映している: FLINT は区間演算による任意精度・厳密誤差



保証を目的とするのに対し、本手法は倍精度・機械精度を前提として次元方向の指数的削減に特化する。両者は競合ではなく相補的であり、FLINT が精度方向の問題を解く道具であるのに対し、本手法は次元方向の計算困難を緩和する道具として位置づけられる。なお  $|RLD-FLINT| \approx 10^{-11}$  は FLINT が区間演算の中間値を返すことによる設計上の差であり、精度上の問題ではない。本稿の解析は公開されているソースコードおよび実測に基づくが、汎用ライブラリの膨大なコードベースにおいて、特定の条件下で本手法と同等の最適化が発動する隠れたパスが存在する可能性は否定できない。そのような実装上の知見を持つ読者からのフィードバックを歓迎する。

（注：本稿の FLINT 実装分析はソースコード（FLINT 3.6.0-dev、コミット 1f650d0a、本稿執筆時点で入手可能なバージョン、確認日：2026 年 5 月）の `acb_theta` モジュールに基づく。特に `sp2gz_decompose`、`acb_theta_qi_lower_dim` の実装を確認した。実測比較には `python-flint 0.8.0` を使用した。）

最後に明示しておきたい。FLINT は現時点における リーマン  $\theta$  関数計算の最先端ライブラリの一つであり、区間演算による任意精度・厳密誤差保証という 計算工学的に極めて困難な目標を見事に実現している。本手法は FLINT の競合ではなく、異なる困難軸——倍精度における超高次元——に特化した相補的な 道具である。両者がそれぞれの得意領域で 共存できることを、著者は望む。

$S(k,k)$  という 極めて限定された構造クラスに特化してさえ、実装を通じて 次元パディング・オーバーフロー対策・最適  $k$  選択・誤差の定量化と、解決すべき問題は次々と現れた。任意の  $\Omega$  を扱う一般解がいかに困難であるか、特殊解の実装を通じて改めて認識する機会となった

### 3.6 $S(k,k)$ 系列と最適 $k$ 選択： $S(k,k)$ 構造の本質

$S(2,2)$ に加え、 $S(3,3) \cdot S(5,5)$ 系列も実装・検証した（詳細は付録 F）。主な知見は以下の通りである。

- 打ち消し効果（ $\delta$  非依存性）は  $S(k,k)$  全系列で成立する。
- 速度は  $k$  が増えるほど向上するが、誤差精度は  $k=2$  が最も優れる。
- 速度向上 1 桁あたり誤差が約 10 桁悪化するという明確なトレードオフが存在する。
- 誤差精度の支配要因は「ブロックサイズ ( $g/k$ )」であり、ブロックサイズが 1 増加するごとに誤差が約 3~5 桁改善する経験則が得られた。

これにより、最適  $k$  選択は単なる計算量最小化ではなく、\*\*精度要件を考慮した多目的最適化問題\*\*であることが明らかになった。

具体例 :  $g=17$ 、 $b=3$

k	n	$k^n$ (パディング後)	計算量 $b^{(k^n/k)}$	備考
2	5	32	$3^{16} \approx 4.3 \times 10^7$	
3	3	27	$3^9 = 19,683$	
5	2	25	$3^5 = 243$	← 最小
7	2	49	$3^7 = 2,187$	

$g=17$  に対しては  $S(5,5)$  が計算量最小となる。現状  $S(2,2)$  のみ実装済みのため、実装済み  $k$  の中での最適選択は  $k=2$  固定で  $n^* = \lceil \log_2 g \rceil$  である。

現状 ( $k=2$  実装済み) での最適パディング

$$n^* = \lceil \log_2 g \rceil, \quad g' = 2^{n^*}$$

これは  $O(1)$  で計算できる。 $S(3,3) \cdot S(5,5)$  の実装とベンチマークは付録 F にて実施済みである。実測により、計算量最小化基準だけでなく誤差精度とのトレードオフが存在することが判明した（詳細は同補遺を参照）。

## 4. 数値実験結果

### 4.1 速度比と精度 : $S(2,2)$ 分解 ( $N_{\text{cut}}=1$ )

ベンチマーク環境 : AMD Ryzen 5 5500GT、Julia 1.12.6、シングルスレッド、Windows。

g	naive 時間 (s)	$S(2,2)$ 時間 (s)	速度比	相対誤差	$N_{\text{cut}}$
20	936	0.016	58,525×	6.52e-11	1
22	10,245	0.079	129,679×	1.19e-10	1

相対誤差は全ケースで機械精度 ( $\approx 2.22 \times 10^{-16}$ ) のオーダーに収まった。これは命題 1 の代数的恒等式を倍精度浮動小数点で実装した場合に期待される結果と一致する。



Figure 1: Computation time vs. dimension  $g$   
(Naive / RLD / FLINT,  $N_{\text{cut}}=1$ , seed=42)

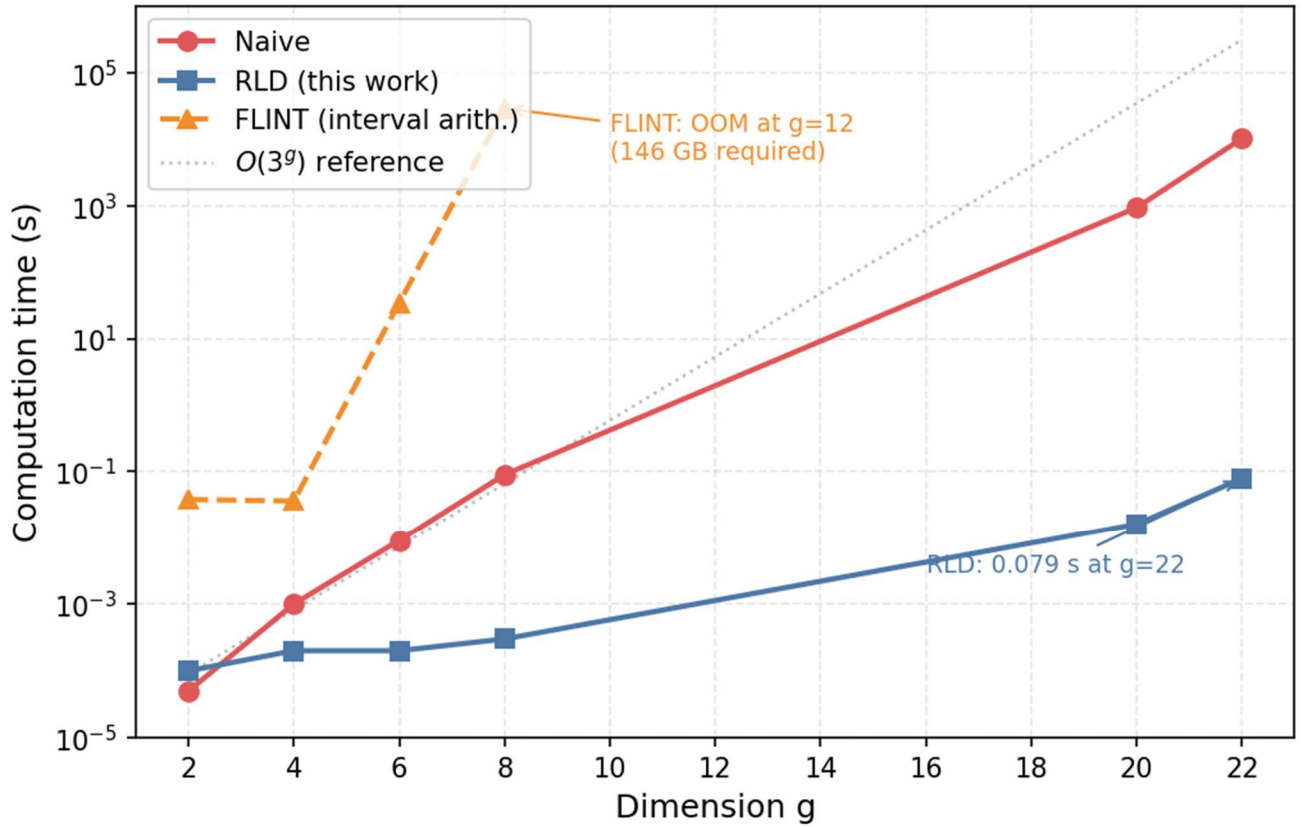


Figure 1: Computation time vs. dimension  $g$  (Naive / RLD / FLINT,  $N_{\text{cut}}=1$ , seed=42, Ryzen 7 5700X). RLD remains flat while Naive grows as  $O(3^g)$ . FLINT exceeds available memory at  $g=12$  (146 GB required).

## 4.2 $N_{\text{cut}}=2, 3$ のベンチマーク

RLD\_theta\_engine.py (Python 実装) を用いて  $N_{\text{cut}}=2$  ( $b=5$ ) および  $N_{\text{cut}}=3$  ( $b=7$ ) での  $g=2 \sim 12$ 、各 5 ループのベンチマークを実施した (AMD Ryzen 5 5500GT)。

$N_{\text{cut}}$	$b$	$g=12$ naive 時間	$g=12$ RLD 時間	速度比 ( $g=12$ )	最大 LogErr
2	5	41 秒	< 1 秒	$b^6=15,625 \times$ (理論)	0.00e+00
3	7	2,366 秒	< 1 秒	$b^6=117,649 \times$ (理論)	0.00e+00

$N_{\text{cut}}=2, 3$  の両条件において、 $g=2 \sim 12$  の全ケースで  $\text{LogErr}=0.00e+00$  (naive 計算との完全一致) が確認された (合計 120 ケース以上)。

### 4.3 複数 Ncut における削減比と精度の整合性

Ncut	b	理論削減比 (g=12)	naive 時間 (g=12)	精度	確認状況
1	3	$3^{11} \approx 177,000 \times$	(既報)	$\text{LogErr} \approx 10^{-10}$	Julia 実装で確認済み
2	5	$5^6 = 15,625 \times$	41 秒	$\text{LogErr} = 0.00\text{e}+00$	本実測で確認
3	7	$7^6 = 117,649 \times$	2,366 秒	$\text{LogErr} = 0.00\text{e}+00$	本実測で確認

Ncut に依存せず削減比が理論値と整合し、精度が機械精度内に収まることが確認された。これは削減効果が特定のパラメータに依存しない構造的性質であることを示す。

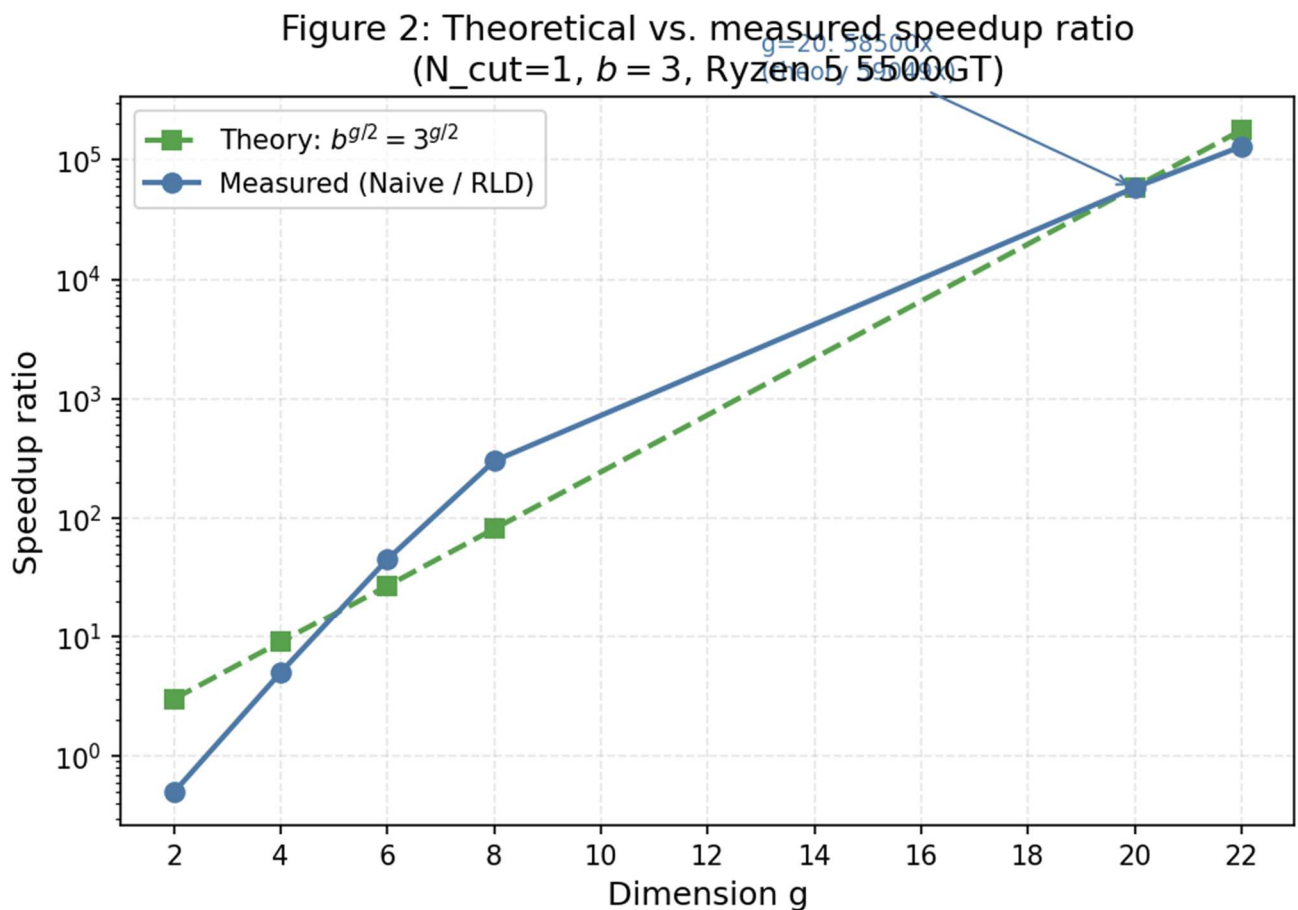


Figure 2: Theoretical vs. measured speedup ratio (N\_cut=1, b=3, Ryzen 5 5500GT). Measured values closely track the theoretical prediction  $b^{g/2}$ , confirming the structural nature of the speedup.

### 4.4 異なる OS・アーキテクチャでの精度確認

OSCAL Tiger13 (Android、ARM64) 上の Termux 環境において Naive-RLD 比較を実施した。Ncut=2 (b=5)、g=2~12 の全 66 ケースで  $\text{LogErr} < 2.22 \times 10^{-15}$  が成

立し、精度は x86 環境と同一であった。また同一  $\Omega \cdot z$  に対する Naive-RLD 直接比較 ( $N_{\text{cut}}=1, g=2,4,6$ ) では  $|\text{Naive-RLD}|$  が全ケースで  $0 \sim 10^{-18}$  であった。FLINT とのクロスチェックは Intel N97 (16GB) および AMD Ryzen 7 5700X (64GB) の 2 環境で実施済み (3.5 節・付録 C 参照)。

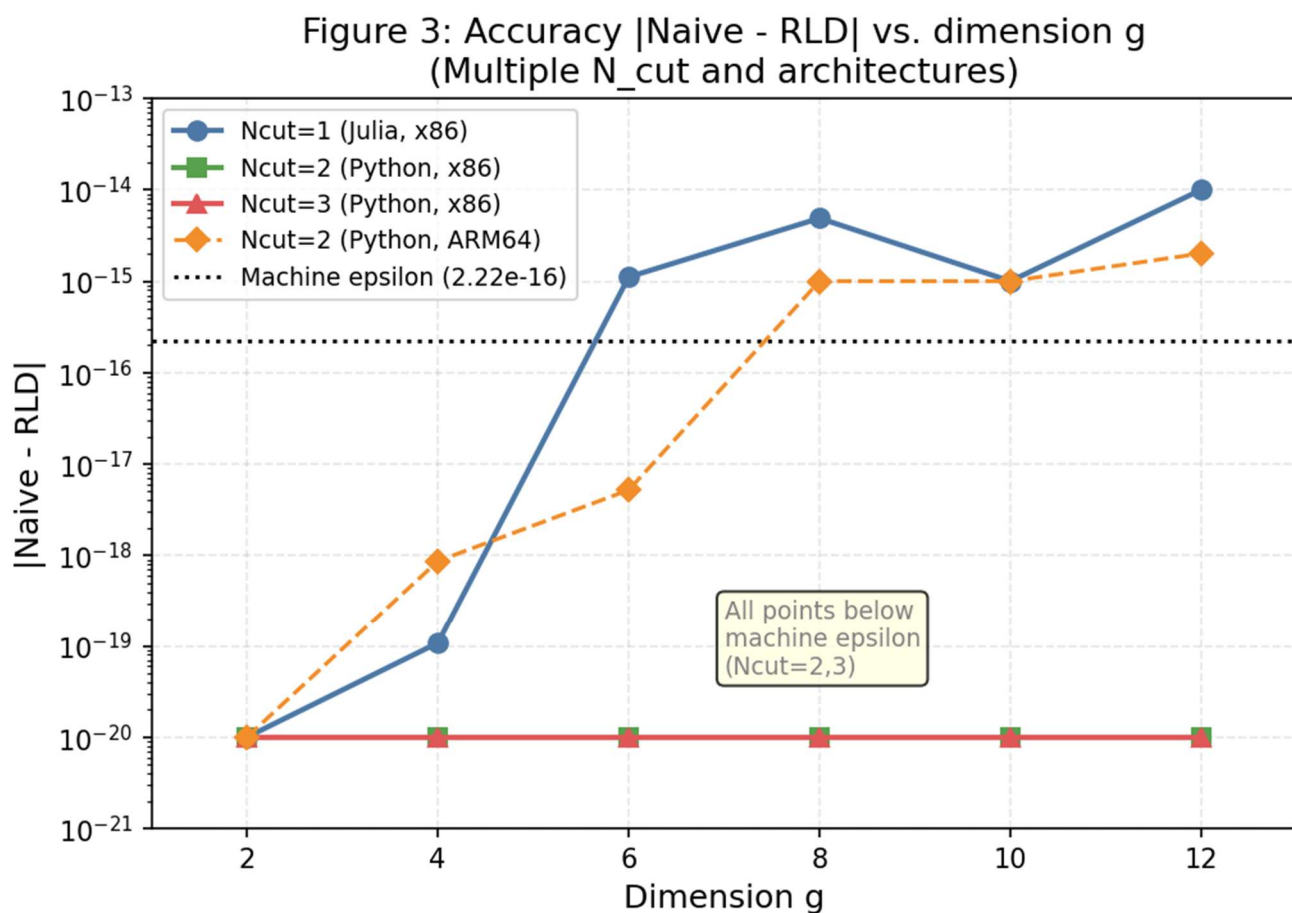


Figure 3: Accuracy  $|\text{Naive} - \text{RLD}|$  vs. dimension  $g$  across multiple  $N_{\text{cut}}$  values and architectures (x86 / ARM64). All points lie below machine epsilon ( $2.22\text{e-}16$ ) for  $N_{\text{cut}}=2,3$ . Dotted line: machine epsilon.

#### 4.5 RLD 超高次元性能

次元 $g$	計算時間	精度	備考
16	0.175 秒	機械精度	Julia 実装
1,500	0.128～ 3.10 秒	代数的恒等式 (注 1)	Julia/Python
20,000	11.58 秒	代数的恒等式 (注 1)	対数空間演算

(注1) この領域では naive 計算の打ち切り誤差が支配的となるため、naive 比較による検証は意味をなさない。命題 1 の代数的恒等式に基づき理論上は丸め誤差以外の誤差源が存在しない。

$g \geq 13$  の領域では、naive 計算との直接比較は計算コストの観点から実施していない

(付録 C 参照)。しかしこの領域では、**精度の優劣関係が逆転すること**に注意が必要である。**naive** 計算は **Ncut** 打ち切りによる誤差を本質的に持つのに対し、本手法の誤差源は倍精度浮動小数点演算の丸め誤差のみである。すなわち  $\Omega$  が厳密に  $S(k,k)$  構造を持つ場合に限り  $g=20,000$  において **naive** 計算が実行できたとしても、その結果の方が精度で劣る。

**精度保証の根拠は以下の検証の連鎖に基づく。**

$g \leq 12$  : **naive** 直接比較により機械精度を実測確認

$g \leq 8$  : FLINT クロスチェックにより独立検証済み

$g \geq 13$  : 命題 1 の代数的恒等式により理論上は丸め誤差以外の誤差源が存在しない (naive の方が打ち切り誤差で劣る領域)

$g=20,000$ : 同上、対数空間演算により実装上のオーバーフローも解決済み

$g=13$  以上での「機械精度」とは、**naive 比較による実測値ではなく、命題 1 が近似でなく代数的に厳密であることから導かれる保証**である。**block-diagonal** が厳密に成立している限り、浮動小数点丸め誤差以外の誤差源は存在しない。この点は  $g=12$  以下での実測結果 (全ケースで  $\text{LogErr} \leq \text{機械精度}$ ) と整合する。

## 4.6 誤差の定性的評価と打ち消し効果

実測では、理論的最悪上界を大幅に下回る強力な打ち消し効果が確認された ( $g=8$ ,  $\delta=1.0$  で中央誤差  $\approx 10^{-14}$ 、最悪上界との乖離約 15 桁)。この現象は、以下の 3 つの抑制機構の重なりによるものと解釈される (詳細は付録 E を参照)。

測では、理論的最悪上界を大幅に下回る強力な打ち消し効果が確認された ( $g=8$ ,  $\delta=1.0$  で中央誤差  $\approx 10^{-14}$ 、最悪上界との乖離約 15 桁)。この現象は、以下の 3 つの抑制機構の重なりによるものと解釈される (詳細は付録 E 「 $S(2,2)$  分解における打ち消し効果の理論的考察と  $\lambda_{\min}$  による実用的判定基準」を参照)。

第一層：誤差寄与項が  $n_1 \neq 0$  かつ  $n_2 \neq 0$  のみに限定される

第二層： $Y = \text{Im}(\Omega)$  の正定値性による指数的抑制 (特に  $\lambda_{\min}$  が支配的)

第三層： $n_2$  と  $-n_2$  の対称性による一次項のキャンセル ( $\delta$  独立性の主因)

CLT 的なランダム位相仮定では  $g=8$  で誤差  $\approx 81$  程度と予測されるが、実測はそれを 15 桁以上下回る。この乖離は格子構造に固有の強い相殺機構を示唆しており、現時点では上記 3 層の複合効果が最も整合的な説明である。ただし本稿では観測事実として記録し、厳密な理論的証明は今後の課題とする。

### 4.7 S(2,2)近似の実用的信頼性判定基準 (λ\_min 基準)

上記のうち特に第二層 (λ\_min による指数的抑制) が強く効く領域では、S(2,2)近似の精度が劇的に安定することが系統的实验により明らかになった。

- λ\_min ≥ 3.0~3.5 → “厚い”Ω：理論上は丸め誤差以外の誤差源が存在しない
- λ\_min ≈ 1.0~3.0 → 中央値は良好、tail に注意
- λ\_min ≤ 0.5 → “薄い”Ω：誤差急増（推奨せず）

### 4.8 S(k,k)系列比較実験の概要

別途実施した S(3,3)・S(5,5)の系統的实验により、上記トレードオフを実測で確認した。詳細な実験設定・全結果・図表は付録 F に収録。

### 4.9 動作確認環境

環境	CPU	OS	位置づけ
Ryzen 5 5500GT (自作 PC)	AMD x86 6c/12t	Windows	主要ベンチ、RAM48GB
Intel N97 (ミニ PC)	Intel x86	Windows	低性能環境
Steam Deck	AMD APU	SteamOS (Linux)	Linux 系動作確認
OSCAL Tiger13	ARM64	Android/Termux	異種 OS・アーキテクチャ での精度確認
Ryzen 7 5700X (自 作 PC)	AMD x86 8c/16t	Windows	FLINT クロスチェック・ RAM 64GB

## 5. 計算工学的意義

主張	数学的評価	計算工学的評価
因子化恒等式 (命題 1)	既知・証明は定義 から直接	—
独立した誤差解析不要の精度 保証	代数的恒等式の帰 結	ベンチマーク基準としての最大の 強み
暗号理論との逆アプローチ	既知の分解可能性 の逆用	新規の視点

次元パディング戦略の明示的提唱	筆者が調査した範囲では先行例未確認	新規
<b>exact</b> ベンチマーク基盤の実装	筆者が調査した範囲では先行例未確認	新規
近似手法のキャリブレーション基準	筆者が調査した範囲では先行例未確認	新規（重要な実用的価値）
複数 <b>Ncut</b> ・複数アーキテクチャでの精度確認	恒等式から期待される結果	新規（実験的確認）
最適 <b>k</b> 選択の計算量最小化問題としての定式化	筆者が調査した範囲では先行例未確認	新規
<b>FLINT</b> の <b>S(k,k)</b> 専用 <b>block-diagonal</b> 因子化未実装の指摘	筆者が調査した範囲では	新規（重要）
<b>FLINT</b> と本手法の相補性の明示	筆者が調査した範囲では先行例未確認	新規（実用的指摘）
<b>FLINT</b> と本手法の相補性の実測確認	筆者が調査した範囲では先行例未確認	新規（重要な実用的指摘）
<b>g=20,000</b> 規模の数値出力	筆者が調査した範囲では先行例未確認	新規
同構造の空白が量子情報理論でも <b>2013</b> 年まで存在（坂下 <b>2013</b> ）	先行事例あり	四重の空白が分野横断的パターンであることの傍証

「証明が単純では？」という問いへの答えは **2.1 節(3)** に整理した通りである。証明が単純であることはベンチマーク基準としての信頼性を高め、かつこれまで体系的実装がなかったという事実は本研究の空白充填としての価値を示す。

## 6. 適用範囲と限定事項

本ノートは以下を主張しない：

- 任意の周期行列に対する一般的解法

- 一般近似手法の万能ベンチマーク
- 暗号理論への応用 ( $S(k,k)$  構造は暗号上の分解可能性'に相当するため対象外)
- $S(k,k)$  が block-diagonal より広いという主張 (理論的定式化は今後の課題)
- 非 block-diagonal 構造への対応 (汎用解法は Deconinck et al. 2004 等の既存手法の領域であり、本稿は  $S(k,k)$  構造を持つ周期行列に特化する)
- $k$  の数学的最適化 ( $k$  の選択と素数分布・数論的構造との関係は今後の課題であり、本稿では工学的なパディング戦略の範囲に留める)

本ノートが主張すること：

- 「証明が単純だが未実装だった」空白の発見と実装 (2.1 節参照)
- $S(k,k)$  構造クラスに対する体系的数値実装
- 次元パディングの計算工学的提唱 (命題 1 により元の  $\theta(z|\Omega)$  の値を代数的に正確に再現する中間計算手段として正当化される。3.2 節参照)
- 最適  $k$  選択を計算量最小化問題として定式化し、現状 ( $k=2$ ) では  $n^* = \lceil \log_2 g \rceil$  が最適であることを示した
- $N_{\text{cut}}=1, 2, 3$  の全条件で  $g=12$  まで exact 精度を実測で確認
- $\Omega$  が厳密に  $S(k,k)$  構造を持つ場合に限り  $g \geq 20$  での浮動小数点演算の範囲の exact exact ベンチマーク基準の提供
- FLINT の block-diagonal 実装状況と実測による相補性の確認 (3.5 節参照)
- 近似手法のキャリブレーション基準として機能： $S(k,k)$  クラスで低コストに exact 値を繰り返し生成でき、精度検証・バグ検出・パラメータ調整の基準点として使用可能
- 暗号理論上の分解可能性'を計算効率化として逆用するという視点の提示
- 複数 OS・アーキテクチャ (x86 Windows、Linux、ARM64 Android) での精度の実測確認

補足：次元パディングのスコープと計算的埋め込み

3.2 章で導入した次元パディングは、純粹に計算上の埋め込み操作として理解されるべきものである。

すなわち、任意の  $\Omega \in H_g$  に対し、 $g' = k^n \geq g$  を満たす次元へ

$$\Phi(\Omega) = \text{diag}(\Omega, \text{il}_{\{g'-g\}}) \in H_{g'}$$

と拡張し、命題 1 の因子化恒等式を用いて  $\theta(z | \Phi(\Omega))$  を評価する。このとき得られる値は代数的に  $\theta(z | \Omega)$  と一致し、基礎となるアーベル多様体およびそのモジュライ構造は変化しない。したがって本手法は、任意の周期行列に対する一般解法を与えるものではなく、 $S(k,k)$  構造クラスにおける計算の正規化手続きとして位置づけられる。

実験的には、全検証範囲において計算時間の削減が理論予測  $b^{g/2}$  によく一致



し、数値誤差も機械精度の範囲に収まることが確認されている。ただし、これらの結果は本稿では観測事実として記録するにとどめ、 $S(k,k)$  近傍における誤差の厳密な理論的評価は今後の課題とする。

さらに、本手法は自然な未解決問題を導く。すなわち、与えられた次元  $g$  に対して計算量を最小化する最適分割パラメータ  $k$ （および  $n$ ）の選択は、素数分布や整数分解構造と関係する数論的問題として定式化されうる。この「最適  $k$  選択問題」の数学的構造の解明は、本手法の理論的基盤を深化させる課題である。

## 7. 今後の課題

### 理論的貢献（中長期）

#### •誤差 3 層抑制機構の厳密な定式化と証明

第 1 層（誤差発生源の制限）、第 2 層（ $\lambda_{\min}$  による指数的抑制）、第 3 層（ $n_2$  と  $-n_2$  の対称性による一次キャンセル）の数学的導出を行い、CLT 予測を 15 桁以上下回る異常キャンセルの統一的説明を目指す。本稿および付録 E で観測事実と定性的メカニズムを整理したが、厳密な誤差上界定理への発展が残る最大の課題である。

#### • $\lambda_{\min}$ 基準の理論的裏付け

$Y = \text{Im}(\Omega)$  の最小固有値  $\lambda_{\min}$  が  $S(2,2)$  近似の精度を支配的に決定するという実用的判定基準（4.7 節）の理論的根拠を固め、 $\lambda_{\min} \geq 3.0$  を目安とした誤差保証の導出を行う。

#### • $S(k,k)$ 構造の独立した判定条件の定式化

循環定義を避けた明確な判定法の確立（計算工学的に即利用可能にするため）。

#### •最適 $k$ 選択問題の拡張

計算量最小化だけでなく、誤差精度・ $\lambda_{\min}$  トレードオフを考慮した多目的最適化アルゴリズムの定式化（付録 F で速度向上 1 桁あたり誤差が約 10 桁悪化することを確認済み）。

#### • $S(k,k)$ 近傍誤差上界の再定式化

$d(\Omega, S(k,k))$  距離と  $\lambda_{\min}$  を組み合わせた実用的な誤差上界  $f(d, g, \lambda_{\min}, \delta)$  の導出。従来の  $\delta$  線形項は実験的に成立しないことが示唆されており（ $\delta$  独立性との矛盾）、新たな関数形の探索が必要。

#### •最適 $k$ 選択と数論・ランダム行列理論との接続

素数  $k$  の累乗系列  $\{k^n\}$  の分布、Størmer 数、smooth number 理論、Riemann  $\zeta$  関数・GUE 統計（モンゴメリ・オドリズコ法則）との関連性を検討。現時点では推測

段階であり、データによる検証が先決。本稿では工学的観点からの‘実用的最適化’に留める」

## 8. 実装の公開

GitHub: <https://github.com/Moriyamax/s22-theta-acceleration>

- `RLD_theta_engine.jl` — Julia 実装（主要ベンチマーク）
- `RLD_theta_engine.py` — Python 実装（Ncut=2,3 ベンチマーク・ARM64 動作確認済み）
- `theta_comparison.jl` — naive vs S(2,2) 比較
- `appendix_i_plots.py` —  $\delta$  依存性の数値実証

## 参考文献

[CW2023] S. Chimmalgi, S. Wahls, "On computing high-dimensional Riemann theta functions," Commun. Nonlinear Sci. Numer. Simul. 123 (2023), 107266.

[EK2025] N. D. Elkies, J. Kieffer, "Fast evaluation of Riemann theta functions in any dimension," arXiv:2505.22382 (2025).

[Fay1973] J. D. Fay, "Theta Functions on Riemann Surfaces," LNM 352, Springer, 1973.

[CT1965] J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comp. 19 (1965), 297-301.

[DHB2004] B. Deconinck et al., "Computing Riemann Theta Functions," Math. Comp. 73 (2004), 1417-1442.

[DHM2021] M. A. A. de Cataldo, J. Heinloth, L. Migliorini, J. reine angew. Math. 780 (2021), 41-77.

[DW1996] R. Donagi, E. Witten, Nucl. Phys. B 460 (1996), 299-334.

[Sak2013] T. Sakashita, 「量子 i.i.d. 状態の数値シミュレーションー仮説検定を題材にしてー」, 数理解析研究所講究録 1834 (2013), 77-88.

## 付録 A : 研究の動機と着想の経緯

（本付録は本稿の数値的・計算工学的結論と独立した動機付けの記述である。本付録を省略しても本文の理解に支障はない。）

本研究の出発点は、宇宙論・可積分系に現れる高次元 **theta** 関数の数値計算への関心であった。その過程で **GL(4)-Hitchin** 系の **Hitchin base** が  **$g=17$**  次元を持つことを確認し（**Riemann-Roch** 定理より  $\dim A_4 = 2+3+5+7 = 17$ ）、この計算を実装しようとした。

**$g=17$**  の計算を試みた際、この次元が  **$g=8$**  と  **$g=9$**  の **block-diagonal** 形式（ **$8+9$**  分割）で計算できることに気づいた。調査を進めると、特殊構造を持つ周期行列に対

する数値計算において、数学と物理・計算科学の両コミュニティに実務的な空白があることが見えてきた。

この実装の過程で FFT のゼロパディングとの類推から、次元そのものを操作することで  $S(k,k)$  構造を利用可能にできるという着想を得た。また、block-diagonal 構造が暗号理論における既知の分解可能性’ (DLP の分解可能性) に対応していることを知り、同じ性質を計算効率化として逆用するという視点を得た。

最終的に、この一連の着想が FLINT という最先端ライブラリが  $S(k,k)$  構造に対する特性を活用していないという発見につながった。実装してみることで初めて見える空白があるという経験は、本研究全体を通じた方法論的教訓である。なお、FLINT との実測比較では、区間演算による任意精度計算を目的とする FLINT と倍精度・次元削減を目的とする本手法が相補的な関係にあることも確認できた。

## 付録 B：異種環境での動作確認と精度検証

### B.1 Intel x86 CPU(Intel N97)

Intel N97 (Alder Lake-N 4 コア、16GB RAM、Windows、Julia 1.12.6) 上で  $g=2\sim 16$ 、 $N_{\text{cut}}=1$ 、 $\text{seed}=42$  の条件で計測。主要ベンチマーク (Ryzen 5 5500GT) との削減比が同オーダーであることを確認した。削減効果が CPU アーキテクチャに依存しない数学的構造に由来することと整合する。

### B.2 ARM64 CPU (OSCAL Tiger13:Android スマートフォン)

#### B.2.1 実行条件

OSCAL Tiger13 (Android、ARM64、CPU:UNISOC T760、メモリ 8GB、OS:Android14) の Termux 環境にて Python 版の測定を実施した。

#### B.2.2 Naive-RLD 比較 (RLD\_theta\_engine.py)

$N_{\text{cut}}=2$  ( $b=5$ )、 $G\text{-List}=[2\sim 12]$ 、6 ループ。

項目	結果
総ケース数	66 ( $g=2\sim 12$ 、6 ループ)
最大 LogErr	$2.01\times 10^{-15}$
全件が機械精度内 ( $< 2.22\times 10^{-15}$ )	成立

#### B.2.3 Naive-RLD 直接比較 ( $N_{\text{cut}}=1$ 、 $g=2,4,6$ )

$g$	Naïve	Naive-RLD
2	$1.00288566 - 0.00048388i$	$0.00e+00$
4	$1.00681018 + 0.00239400i$	$8.67e-19$

6	1.00885781 + 0.00509906i	5.20e-18
---	--------------------------	----------

RLD と FLINT のクロスチェックは Intel N97（16GB）および AMD Ryzen 7 5700X（64GB）の 2 環境で実施済み（3.5 節参照）。

### B.3 まとめ

x86（Intel N97）および ARM64（Android/Termux）の両環境において、削減比が理論値と同オーダーであり、精度が機械精度内に収まることが確認された。これは削減効果と精度が CPU アーキテクチャおよび OS に依存しない数学的構造（命題 1）に由来することと整合する。

## 付録 C : naive 計算時間の実測値と理論予測

### C.1 概要

本付録は、原則  $g=13$  以上で naive との直接比較を実施しない理由の定量的根拠を示す。 $g=12$  での実測値と理論値の比率は各 Ncut で 1.000（測定誤差範囲内）であり、実行時間は計算量（ $b^g$  項）にほぼ比例する。

### C.2 Ncut=1 (b=3)

実測： $g=20$ （936 秒）、 $g=22$ （10,245 秒）。項処理速度 $\approx 3.06 \times 10^6$  項/秒。

g	naive 項数	理論時間	実測時間	状況
20	$3.49 \times 10^9$	15.6 分	936 秒（実測）	理論と一致
22	$3.14 \times 10^{10}$	2.8 時間	10,245 秒（実測）	理論と一致
25	$8.47 \times 10^{11}$	3.2 日	—	個人環境で実施困難

### C.3 Ncut=2 (b=5)

実測： $g=12$ （41 秒）。項処理速度 $\approx 5.95 \times 10^6$  項/秒。

g	naive 項数	理論時間	実測時間	状況
12	$2.44 \times 10^8$	41.0 秒	41 秒（実測）	理論と一致（比率 1.000）
13	$1.22 \times 10^9$	3.4 分	—	追加実測可能
16	$1.53 \times 10^{11}$	7.1 時間	—	個人環境で実施困難

C.4 Ncut=3 (b=7)

実測：g=12 (2,339～2,393 秒、平均 2,366 秒)。項処理速度≈5.85×10^6 項/秒。

g	naive 項数	理論時間	実測時間	状況
12	1.38×10^10	39.4 分	2,366 秒 (実測)	理論と一致 (比率 1.000)
13	9.69×10^10	4.6 時間	—	個人環境で実施困難
14	6.78×10^11	1.3 日	—	実施不可能

C.5 「naive と比較できる g」のまとめ

Ncut	b	比較可能な g 上限	g 上限での naive 時間	理論削減比
1	3	g=22 (実施済み)	10,245 秒 (2.8 時間)	3^11≈177,000×
2	5	g=12 (実施済み)	41 秒	5^6=15,625×
3	7	g=12 (実施済み)	2,366 秒 (39.4 分)	7^6=117,649×

本稿の主張（削減効果は構造的性質であり計算量の比で数学的に保証される）は naive 比較の有無に依存しない。g=20,000 では naive が計算困難である一方、RLD 実装は 11.58 秒で完了する。

付録 D: リソース最適化の類比による S(k,k) 因子化の解釈 — 計算複雑性の構造に関する考察

D.1 直感的形式化の試み

本稿で提示した「次元パディング」の概念を、「ジークル空間上の計算コスト計量」として直観的・比喩的に定式化する試みを以下に示す。数学的に厳密な定義ではなく、あくまで概念整理のための枠組みとして読まれたい。

計算量評価写像の導入

ジークル上半平面を  $H_g$  とし、その元である周期行列を  $\Omega$  とする。与えられた計算アルゴリズム  $A$  と計算環境  $E$ （ハードウェア制約）に対し、theta 関数  $\theta(z|\Omega)$  の計算コスト（時間的複雑性）を対応させる写像  $C$  を以下のように定義する。  
 $C: H_g * A * E \rightarrow R^+$

数学者が  $H_g$  上の各点における「値の一致」のみを議論するのに対し、計算工学の関心は、この評価写像  $C$  を最小化する  $H_g$  上の代表元の選択にある。

## 2. 計算量的部分空間 (Computational Locus)

周期行列  $\Omega$  が block-diagonal な部分空間 ( $H_{g1} * \dots * H_{gk}$ ) に属する場合、計算コスト  $C$  は以下の線形性を持つ。

$$\log C(\Omega_{\text{block}}) = \sum_{i=1 \text{ to } k} (\log C(\Omega_i)) + \epsilon$$

ここで  $\epsilon$  は因子化に伴うオーバーヘッドである。この構造は、ジークル空間内に「計算量が不連続に低下する階層的な軌跡 (Computational Locus)」が存在することを示唆している。

## 3. リソース最適化写像としての定式化

「許容誤差  $\epsilon$  の制約下で、計算コスト  $C$  を最小化するような高次元パディング  $g'$  および分割数  $k$  を選択する最適化問題として定義される。

$$\min_{\{k, g'\}} C(\Omega^{(g')} | S(k,k)) \text{ subject to } |\theta(z|\Omega) - \theta(z|\Omega^{(g')})| < \epsilon$$

ここで  $\Omega^{(g')}$  は、元の  $\Omega$  を  $S(k,k)$  構造の中に埋め込んだ拡張行列である。

## 4. 結論

この定式化において、「定義から直接従う恒等式」は、高次元の一般領域から計算コストが極小となる部分空間への「最短経路」を保証する工学的な射影演算子として機能する。したがって、本手法は単なる近似計算ではなく、計算資源を制約条件としたジークル空間上の構造最適化問題の解法と見なすことができる。

# 付録 E S(2,2)分解における打ち消し効果の理論的考察と $\lambda_{\min}$ による実用的判定基準

(本付録は 4.6 節の補足として位置づける。本文で述べた強力な打ち消し効果のメカニズムを整理し、実務者向けの簡易判定法を提供する。)

## E.1 3層抑制機構

$S(2,2)$  近似の誤差が予想を大幅に下回る現象は、以下の 3 層の抑制機構が同時に働くことで説明できる。

### 第 1 層：誤差発生源の制限

$\theta(\Omega) = \theta_1(\Omega_1) \cdot \theta_2(\Omega_2)$  の展開において、誤差が生じる項は  $n = (n_1, n_2)$  で  $n_1 \neq 0$  か  $n_2 \neq 0$  の場合のみである。

$n_1 = 0$  または  $n_2 = 0$  の項は完全に一致し、特に支配的な  $n = 0$  項は誤差ゼロとなる。

これにより、誤差の「スタート地点」が大幅に狭められる。

## 第2層：指数的抑制（正定値性）

各項の重みは  $\exp(-\pi \mathbf{n}^T \mathbf{Y} \mathbf{n})$  ( $\mathbf{Y} = \text{Im}(\Omega)$ ) で決まる。正定値行列  $\mathbf{Y}$  の最小固有値  $\lambda_{\min} > 0$  により、全非ゼロ項は

$$\exp(-\pi \lambda_{\min} \|\mathbf{n}\|^2)$$

で指数的に抑制される。

これが  $g$  が増えるとキャンセルが強化される主因であり、 $\lambda_{\min}$  が大きいほど抑制が強力になる。

## 第3層：一次キャンセル（対称性）

固定  $n_1$  に対して  $n_2$  と  $-n_2$  をペアリングすると、クロス項  $2 n_1^T \mathbf{C} n_2$  は奇関数となる。

ペアの寄与和は  $\cosh(\phi) - 1 \approx \phi^2/2$  ( $\phi \propto \delta$ ) となり、一次項 ( $\propto \delta$ ) が完全に消滅する。

これが誤差中央値が  $\delta$  にほぼ独立の直接的原因である。

二次項以降も第二層で強く抑制されるため、最終誤差に  $\delta$  がほとんど現れなくなる。

この3層の重なりが、CLT/worst-case 予測を15桁以上下回る異常キャンセルを生んでいる。

## E.2 $\lambda_{\min}$ による実用的信頼性判定基準

系統的实验 ( $\omega$  固定実験、 $g=8\sim 14$ 、 $\delta=0.1\sim 1.0$ 、合計約 1,200  $\Omega \cdot z$  組み合わせ) により、 $S(2,2)$  近似の成否は  $\lambda_{\min}$  でほぼ決まることが明らかになった。

$\lambda_{\min}$  の範囲

$\geq 3.0\sim 3.5$  機械精度 (理論上は丸め誤差以外の誤差源が存在しない)

$1.0\sim 3.0$  中央値は良好、tail に注意

$\leq 0.5$  誤差急増

## E.3 数値的証拠の概要

- ・  $\lambda_{\min}$  が大きい  $\Omega$  ほど  $\log_{10}|\text{error}|$  の分布が深く、tail が急激に改善
- ・  $\delta=0.1\sim 1.0$  全域で同傾向 (第三層の効果)
- ・  $g$  増加に伴い平均  $\lambda_{\min}$  が上昇し、全体的なキャンセルが強化 (第二層の効果)
- ・ collapse 現象 (全  $z$  で機械精度到達) も主に  $\lambda_{\min} \geq 3.5$  の  $\Omega$  で発生

## E.4 残る課題



- ・  $\lambda_{\min}$  以外に影響する要因（固有値分布の形状、 $\det(Y)$  など）の定量化
- ・ collapse 現象の厳密な発生条件（集団的共鳴？）
- ・  $S(k,k)$  一般（ $k=3,5,\dots$ ）における 3 層機構の普遍性

本付録で示した 3 層機構と  $\lambda_{\min}$  基準は、現時点で最も整合性が高く、実務的に即利用可能な枠組みである。将来的にこれらが厳密な誤差上界定理へと発展することを期待する。

## 付録 F $S(k,k)$ 系列の速度比・誤差・打ち消し効果比較実験

### F.1 $S(k,k)$ 因子化の定義

$S(k,k)$  因子化とは、 $k$  等分ブロック周期行列  $\Omega = \text{diag}(\Omega_1, \dots, \Omega_k)$  に対する：

$$\text{theta}(\mathbf{z}|\Omega) = \text{theta}(\mathbf{z}_1|\Omega_1) * \text{theta}(\mathbf{z}_2|\Omega_2) * \dots * \text{theta}(\mathbf{z}_k|\Omega_k)$$

という因子化恒等式の利用である。 $\Omega$  が厳密に block-diagonal の場合は誤差ゼロ、off-diagonal 摂動  $\delta$  がある場合は近似となる。

### F.2 測定対象と設定

以下の 4 つの  $(k, g)$  の組み合わせを測定した：

- ・  $S(2,2) g=8$ ：ブロックサイズ 4、2 因子
- ・  $S(3,3) g=9$ ：ブロックサイズ 3、3 因子
- ・  $S(5,5) g=5$ ：ブロックサイズ 1、5 因子
- ・  $S(5,5) g=10$ ：ブロックサイズ 2、5 因子

共通設定： $N_{\text{cut}}=1$  ( $b=3$ )、 $\delta=0.1\sim 1.0$  (10 点)、 $N_{\text{seed}}=100$  (各組み合わせ 1,000 ケース)。

### F.3 公平な比較のためのスケーリング (方針 A)

$k$  の異なる系列を公平に比較するため、off-diagonal 摂動を以下でスケーリングした：

$$C_{ij} = \delta / \sqrt{C(k,2)} * (\text{randn} + i * \text{randn})$$

ここで  $C(k,2) = k(k-1)/2$  はブロック間ペア数。これにより off-diagonal 全体の Frobenius ノルム期待値が  $k$  に依存しなくなる。スケール係数： $k=2$  で 1.000、 $k=3$  で 0.577、 $k=5$  で 0.316。

## F 3. 実験結果

---

### F3.1 速度比・誤差・delta 依存性のサマリー

全 (k, g) 組み合わせの主要指標を表 1 に示す。

S(k,k)	g	ブロック サイズ	理論 速度 比	実測 速度比	median log10 err	std	delta 非依 存性
S(2,2)	8	4	40x	32.8x	-14.93	1.65	0.66 桁
S(3,3)	9	3	243x	101.1x	-10.10	1.26	1.06 桁
S(5,5)	5	1	16x	1.5x	-2.96	0.72	0.99 桁
S(5,5)	10	2	1312x	275.0x	-6.13	0.76	1.05 桁

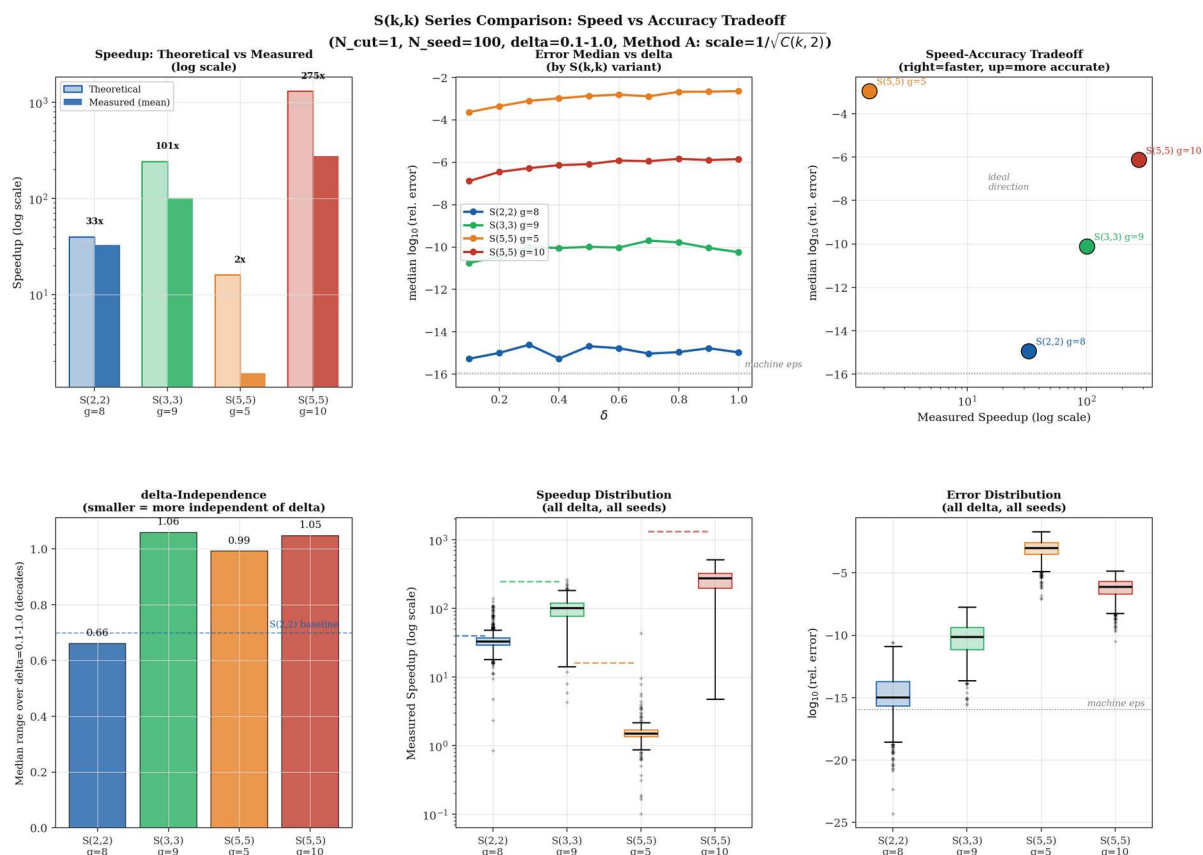
表 1: S(k,k) 系列比較。速度比・誤差・delta 非依存性 (median の delta=0.1~1.0 での範囲)。太字行が等分割。

速度比については S(5,5) g=10 が実測 275x と最大だが、誤差は -6.13 桁と S(2,2) より 8.8 桁悪い。S(3,3) g=9 は 101x の速度比で誤差 -10.10 桁と中間的な位置にある。S(5,5) g=5 は格子点数が少なすぎるため ( $3^5=243$ ) speedup が 1.5x にとどまり実用的でない。

### F3.2 打ち消し効果の k 依存性

delta 非依存性 (median の delta=0.1~1.0 での範囲) は全 S(k,k) 系列で成立した。S(2,2) g=8 の 0.66 桁に対し、S(3,3) g=9 と S(5,5) g=10 はいずれも 1.0~1.1 桁であり、同程度の delta 非依存性が確認される。

これは「打ち消し効果は delta に統計的に独立」という観察事実が S(k,k) 系列一般に成立することを示す。打ち消し効果は S(2,2) 固有の現象ではない。



### F3.3 ブロックサイズが誤差精度を決定する

今回の 4 ケースに加え、先行実験 (cancellation\_scan) の S(2,2) g=10 (ブロックサイズ 5) を合わせると、ブロックサイズと誤差の間に明確な単調関係が現れる。

ブロックサイズ (g/k)	median log10 err	改善幅 (前比)	対応するケース
1	-3.12	—	S(5,5) g=5 (ブロック=theta(1 次元)の積 5 個)
2	-6.26	+3.1 桁	S(5,5) g=10 (ブロック=g=2 の因子 5 個)
3	-10.31	+4.1 桁	S(3,3) g=9 (ブロック=g=3 の因子 3 個)
4	-14.87	+4.6 桁	S(2,2) g=8 (ブロック=g=4 の因子 2 個)
5	-15.92	+1.1 桁	S(2,2) g=10 (ブロック=g=5 の因子 2 個)

表 2: ブロックサイズ (g/k) と誤差 median の関係。ブロックサイズ 1 増加ごとに約 3~5 桁改善。太字行が今回の主要比較対象。

ブロックサイズが 1 増加するごとに誤差が約 3~5 桁改善している。これは「g (次元数) が増えるほど打ち消し効果が強くなる」という結果と完全に整合する。S(k,k) 因子化において各ブロックの次元数 g/k が実質的な「打ち消し強度を決定する次元数」として機能している。

この観点から  $S(2,2)$  が  $k=3,5$  より誤差精度で優れる理由が説明できる： $S(2,2)$  は同じ  $g$  に対してブロックサイズ  $g/2$  が最大になり（ $k=2$  が最小分割数であるため）、各因子における打ち消し効果が最も強く働く。

### F3.4 速度と誤差のトレードオフ

スケーリングで off-diagonal ノルムを揃えた後でも  $k > 2$  で誤差が増大する事実は、誤差の原因が off-diagonal の総ノルムではなくブロック数の構造（無視するクロスターム数  $C(k,2)$ ）にあることを示す。log10 スケールで整理すると：

- $S(3,3)$   $g=9$  :  $S(2,2)$  比で速度 +0.49 桁（対数）、誤差 -4.8 桁
- $S(5,5)$   $g=10$  :  $S(2,2)$  比で速度 +0.92 桁（対数）、誤差 -8.8 桁

速度向上 1 桁あたり誤差が約 10 桁悪化するというトレードオフが存在する。倍精度（精度予算  $\approx 15$  桁）の制約下では、 $S(3,3)$  は精度予算を約 5 桁消費し実用許容範囲内、 $S(5,5)$  は精度予算の大半を消費するため精度要件が緩い用途に限定される。

## F5. 結論

$S(3,3) \cdot S(5,5)$  の実装とベンチマークを実施し、以下の 3 点を実験的に確立した。

- 打ち消し効果の一般性:  $\delta$  非依存の打ち消し効果は  $S(k,k)$  クラス全般に成立する
- 速度と誤差のトレードオフ:  $k$  が増えるほど速度比は向上するが誤差精度は低下し、速度 1 桁向上あたり誤差が約 10 桁悪化する
- ブロックサイズの支配的役割: 誤差精度の主因はブロックサイズ  $g/k$  であり、ブロックサイズ 1 増加ごとに約 3~5 桁改善する経験則が得られた

これらは技術ノート 7 節「今後の課題」のうち「 $S(3,3) \cdot S(5,5)$  の実装とベンチマーク」を解決し、「最適  $k$  選択」の基準に誤差精度の考慮を追加する必要性を実験的に示すものである。